Defense Information Infrastructure (DII) Common Operating Environment (COE)

Version 3.1

Baseline Specifications

April 29, 1997

Joint Interoperability and Engineering Organization
Defense Information Systems Agency

EXECUTIVE SUMMARY

The DII COE concept is best described as an architecture that is fully compliant with the *DoD Technical Architecture for Information Management (TAFIM), Volumes 2 and 3,* and the *DoD Joint Technical Architecture (JTA),* an approach for building interoperable systems, a collection of reusable software components, a software infrastructure for supporting mission area applications, and a set of guidelines and standards. The guidelines and standards specify how to reuse existing software and how to properly build new software so that integration is seamless and, to a large extent, automated.

The COE is a "plug and play" open architecture designed around a client/server model. The COE is *not* a system; it is a *foundation* for building an open system. Functionality is easily added to or removed from the target system in small manageable units called *segments*. Structuring the software into segments is a powerful concept that allows considerable flexibility in configuring the system to meet specific mission needs or to minimize hardware requirements for an operational site. Site personnel perform field updates by replacing affected segments through use of a simple, consistent, graphically oriented user interface.

The COE represents a departure from traditional development programs. It emphasizes incremental development and fielding to reduce the time required to put new functionality into the hands of the warrior while not sacrificing quality nor incurring unreasonable program risk or cost. This approach is sometimes described as a "build a little - test a little - field a lot" philosophy. It is a process of continually evolving a stable baseline to take advantage of new technologies as they mature and to introduce new capabilities. But the changes are done one step at a time so that the warfighters always have a stable baseline product while changes between successive releases are perceived as slight.

The primary purpose of this document is to provide developers with the baseline configuration for the DII COE version 3.1.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1

INTRODUCTION

The Defense Information Infrastructure (DII) Common Operating Environment (COE) originated with a simple observation about command and control systems: certain functions (mapping, track management, communication interfaces, etc.) are so fundamental that they are required for virtually every command and control system. Yet these functions are built over and over again in incompatible ways even when the requirements are the same or vary only slightly between systems. If these common functions could be extracted, implemented as a set of extensible, low-level building blocks and made readily available to system designers, development schedules could be accelerated and substantial savings could be achieved through software reuse. Moreover, interoperability would be significantly improved if common software were used across systems for common functions.

This observation led to the development of the DII COE which is planned for use in systems throughout DoD. Two such systems are the Global Command and Control System (GCCS) and the Global Combat Support System (GCSS). Both systems use the same infrastructure and integration approach and they use the same COE components for functions that are common.

GCCS is a Command, Control, Communications, Computers and Intelligence (C4I) system with two main objectives: the replacement of the World-Wide Military Command and Control System (WWMCCS) and the implementation of the C4I For the Warrior concept. GCCS includes multiple workstations cooperating in a distributed LAN/WAN environment. Key features include "push/pull" data exchange, data processing, sensor fusion, dynamic situation display, analysis and briefing support, and maintenance of a common tactical picture among distributed GCCS sites.

GCSS is presently under development and is targeted for the warfighting support functions (logistics, transportation, etc.) to provide a system that is fully interoperable with the warfighter C4I system. Implemented to its fullest potential, GCSS will provide both warfighter support to include reachback from deployed commanders into the CONUS sustaining base infrastructure and cross-functional integration on a single workstation platform.

Initial COE development was driven by the requirement to build a suitable WWMCCS replacement. WWMCCS maintenance costs were significant and the system was rapidly reaching the point of technical obsolescence. A significant component of the COE challenge was to strategically position the system architecture so as to be able to take advantage of technological advances. At the same time, the system could not sacrifice quality, stability, or functionality already in the hands of the warrior. In keeping with current DoD trends, the COE emphasizes use of commercial products and standards where applicable to take advantage of investments made by commercial industry.

To achieve the WWMCCS replacement objective, technical experts and program managers from each of the services, DODIIS, NIMA, and other interested agencies met for several months beginning in the fall of 1993. Participants proposed candidate systems as a possible starting point for the COE architecture or as a suitable candidate for providing capabilities to meet WWMCCS replacement requirements. None of the candidate systems met all requirements, but it was clear that a combination of the "best" from several systems could produce a system that would be suitable for WWMCCS replacement. Moreover, an infrastructure could be put into place and a migration strategy defined to preserve legacy systems until migration to the intended architecture could be realized.

The cornerstone architectural concept jointly developed during these series of meetings is the DII COE. The present COE is composed of software contributed from several candidate systems evaluated by this joint engineering team. It is being expanded to include global data management and workflow management for GCSS logistics applications. It will expand further as more functional areas employ its services in areas such as Electronic Commerce/Electronic Data Interchange (EC/EDI), transportation, base support, personnel, health affairs, and finance. The COE is described more completely in the *DII COE Integration and Runtime Specifications*. This document also describes technical information required to properly access and extend software contained within the COE.

An initial proof-of-concept system, GCCS 1.0, was created and installed in early 1994 at one operational site to validate the approach and to receive early feedback. GCCS 1.1 followed in the summer of 1994 and was the first attempt to integrate software from the Army AWIS and Navy JMCIS programs as initial COE components. GCCS 1.1 included mission applications from a variety of other programs operating in a "federated" mode, that is, constructed so as to be able to run on the same hardware without interfering with other software, but not yet able to effectively share data between applications. This successful effort allowed GCCS 1.1 to be installed and tested at beta sites and was used at certain operational sites to monitor events during the 1994 Haiti crisis. GCCS 2.0 fielding began in early 1995 at a number of operational sites. GCCS 2.1 was fielded in mid-1995. GCCS 2.2 is scheduled to be released in late-1996. The GCCS 2.0 series marks the real beginning of the DII COE concept. Use of the DII COE is crucial in being able to rapidly integrate software from candidate programs to successfully build a baseline with an ever increasing level of functionality.

The DII COE has its roots in command and control, but the principles and implementation described in this document are not unique to GCCS nor GCSS. The principles and implementation are not limited to command and control or logistics applications, but are readily applicable to many other application areas. The specific software components selected for inclusion in the COE determine the mission areas that the COE can address.

1.1 DOCUMENT SCOPE

This document describes the baseline configuration of the DII COE version 3.1. It will provide an overview of the DII COE version 3.1 architecture, architecture guidelines, configuration, and the associated APIs. This document supersedes the *DII COE Version 3.0 (Series) Baseline Specifications* dated October 31, 1996. All segments submitted to DISA are required to be formatted in accordance with the *DII COE Integration and Runtime Specificiations*.

1.2 APPLICABLE DOCUMENTS AND STANDARDS

This document is one in a series of related documents which define development requirements, system architecture, engineering tools, and implementation techniques. Many of the documents cited are available on the World Wide Web. Alternatively, contact the DII COE Configuration Management office for information on how to obtain the desired documents.

Because the COE and COE-based systems are ongoing programs, enhancements and additional features are developed on a regular basis. Documentation updates are regularly released for each of the documents listed here. Be sure to always reference the latest version for the documents listed below and be aware that many of the documents are being modified and extended to address DII COE-based systems, not just GCCS or GCSS.

DISA, *DII Common Operating Environment Integration and Runtime Specifications version 3.0(Draft),* January 31, 1997. This document describes the technical requirements for using the DII COE to build and integrate systems.

DISA, *DII Common Operating Environment Software Quality Compliance Plan version 3.0,* January 22, 1997. This document provides a description of the metric collection, analysis activities, and schedule. These collection and analysis activities evaluate the level of compliance to the Common Operating Environment (COE) software quality goals. The value of the compliance level assists the DII Engineering Office in understanding the costs and risks of integrating software into the COE to provide common functions. This understanding is used to determine the level of support needed for each COE common function. Specifically, this document describes the metrics collection and analysis process that will ensure the compliance to the software quality guidelines required to provide functionality for the COE.

DoD, Joint Technical Architecture Version 1.0 dated August 22, 1996. The JTA is a document that identifies a common set of mandatory information technology standards and guidelines to be used in all new and upgraded Command, Control, Communications, Computers, and Intelligence (C4I) acquisitions across DoD.

DoD, *Technical Architecture Framework for Information Management*. This is a multi-volume document which defines a standards profile and the DoD Technical Reference Model (TRM) for information management systems.

User Interface Specification for the Defense Information Infrastructure Version 2.0 dated April 01, 1996. This document, sometimes called the *Style Guide*, defines the "look and feel" for developing user interfaces for the DII. This style guide is closely patterned after the commercial Motif style guide.

1.3 DOCUMENT STRUCTURE

Section 1 of this document is an overview of the DII COE.

Section 2 provides a brief description of the DII COE concept in terms of strategy, architecture, and implementation.

Section 3 provides a description of the DII architecture and the DII COE Version 3.1 baseline configuration.

Appendix A provides the current DII COE Taxonomy

Appendix B provides a detailed list, by platform, of the software segments that are included in the DII COE Version 3.1 Kernel.

Appendix C provides a detailed list, by platform, of the DII COE Version 3.1 software segments above and beyond the Kernel.

SECTION 2

THE DII COE CONCEPT

The DII COE concept is a fundamentally new approach that is much broader in scope than simple software reuse. Software reuse itself is not a new idea. Unfortunately, most software reuse approaches to date have been less than satisfactory. Reuse approaches have generally emphasized the development of a large software repository from which designers may pick and choose modules or elect to rebuild modules from scratch. It is not sufficient to have a large repository and too much freedom of choice leads to interoperability problems and duplication of effort. This rapidly negates the advantages of software reuse.

The DII COE does emphasize both software reuse and interoperability, but its principles are more far-reaching and innovative. The COE concept encompasses:

- an architecture and approach for building interoperable systems
- an infrastructure for supporting mission area applications
- a rigorous definition of the runtime execution environment
- a rigorous set of requirements for achieving COE compliance
- an automated toolset for enforcing COE principles and measuring COE compliance
- an automated process for software integration
- a collection of reusable software components
- an approach and methodology for software reuse
- a set of APIs for accessing COE components

In the context of this document, the COE must be understood as a multi-faceted concept. Proper understanding of how the many facets interact is important in appreciating the scope and power of the DII COE and to avoid confusion in understanding COE material. The next subsection deals with three specific facets in more detail: the COE as a system foundation, the COE as an architecture, and the COE as an implementation strategy.

To view the COE as a C4I system is incorrect because it misses the fundamental point that the COE is *not* a system: it is a *foundation* for building an open system. This viewpoint also makes fielding and updating schedules confusing because it fails to account for the impact of the evolutionary development strategy. To view the COE as GCCS or just an architecture gives the mistaken impression that its principles are limited to the GCCS program. GCCS is simply the first system built on top of the DII COE while development of GCSS is still in progress. This view also fails to account for the fact that a baseline already exists composed of components selected from mature service/agency programs. Finally, to view the COE as just an implementation strategy is a limited perspective because it fails to account for the fact that there has been a near-term, real-world objective (WWMCCS replacement). It ignores the evolutionary nature of the COE and mission applications development and it ignores the implied requirement to provide an easy update mechanism for operational sites.

2.1 THE DII COE AS A SYSTEM FOUNDATION

Figure 2-1 shows how the DII COE serves as a foundation for building multiple systems. The shaded box shows two types of reusable software: the operating system and COE components. Section 3 describes the COE components in more detail and the supported operating systems. For the present discussion, it is sufficient to note that these components are accessed through APIs and that they form the architectural backbone of the target system.

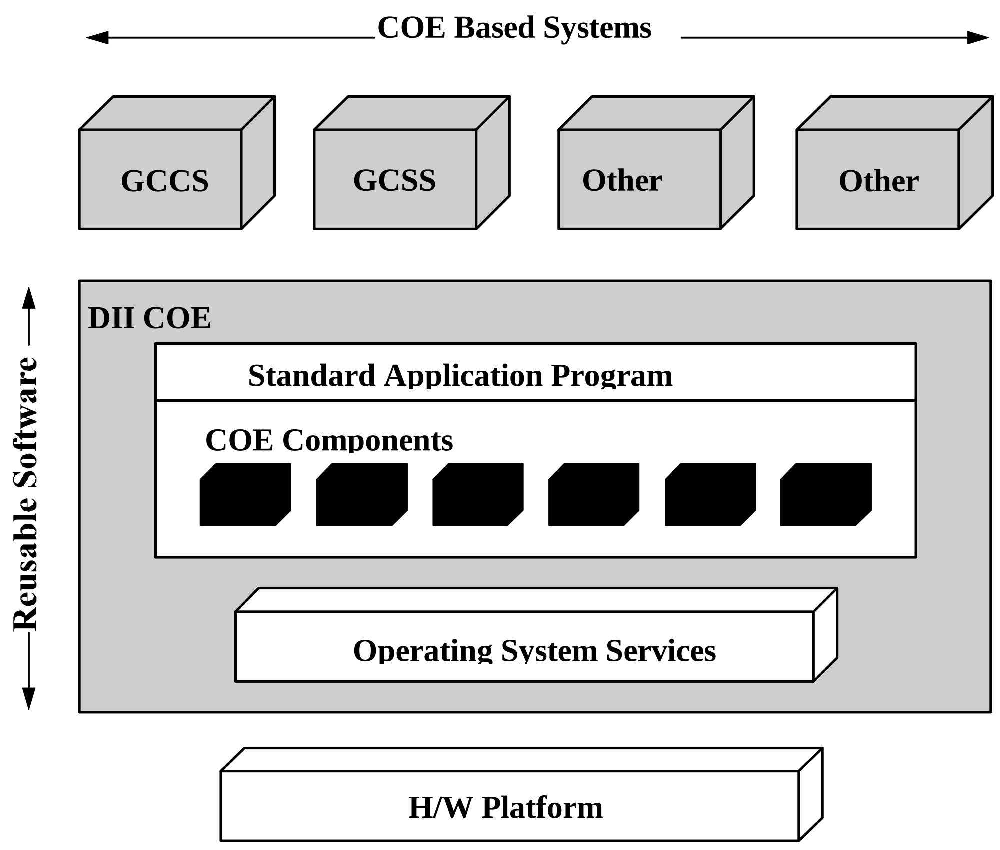

Figure 2-1: DII COE and COE Based Systems

Building a target system such as GCCS or GCSS is largely a matter of combining COE components with mission specific software. The COE infrastructure manages the flow of data through the system, both internally and externally. Mission specific software is mostly concerned with requesting data from the COE and then presenting it in a form that is most meaningful to the operator (e.g., as a pie chart, in tabular form, or as a graph). The COE provides the necessary primitives for such data manipulation and has the necessary information about where the requested data is stored, whether locally or remotely across the LAN/WAN. This frees the system designer to concentrate on meaningful data presentation and not on the mechanics of data manipulation, network communications, database storage, etc.

It must be kept in mind, however, that there is only one COE. Each system uses the same set of APIs to access common COE components, the same approach to integration, and the same set of tools for enforcing COE principles. Systems are built on top of the COE and use precisely the same COE software components, not just the same algorithms, for common functions (e.g., communications interfaces and dataflow management). This approach to software reuse significantly reduces interoperability problems because if the same software is used, there is no chance of interpreting or implementing standards differently.

2.2 The DII COE as an Architecture

The DII COE is a "plug and play" open architecture designed around a client/server model. Functionality is easily added to or removed from the target system in small manageable units, called *segments*. Segments are defined in terms of functions that are meaningful to operators, not in terms of internal software structure. Structuring the software into segments in this manner is a powerful concept that allows considerable flexibility in configuring the system to meet specific mission needs or to minimize hardware requirements for an operational site. Site personnel perform field updates by replacing affected segments through use of a simple, consistent, graphically oriented user interface.

The DII COE model is analogous to the Microsoft Windows® paradigm. The idea is to provide a standard environment, a set of standard off-the-shelf components, and a set of programming standards that describe how to add new functionality to the environment. The Windows paradigm is one of a "federation of systems" in that properly designed applications can coexist and operate in the same environment. But simple coexistence is not enough. It must be possible for applications to share data. The DII COE extends the Windows paradigm to allow for true "integration of systems" in that mission applications share data at the server level.

Federation versus integration is an important architectural advantage. However, integration is not possible without strict standards that describe how to properly build components to add to the system. This document and other related documents detail the technical requirements for a well behaved, COE-compliant application. The COE provides automated tools to measure compliance and to pinpoint problem areas. A useful side effect of the tools and procedures is that software integration is largely an automated process, thus significantly reducing development time while automatically detecting potential integration and runtime problem areas.

More precisely, to a developer the DII COE is:

- **An Architecture:** A precisely defined TAFIM-compliant (Technical Architecture Framework for Information Management), client/server architecture for how system components will interact and fit together and a definition of the system level interface to COE components.

- **A Runtime Environment:** A standard runtime operating environment that includes "look and feel," operating system, and windowing environment standards. Since no single runtime environment is possible in practice, the COE architecture provides facilities for a developer to extend the environment in such a way as to not conflict with other developers.

- **Software:** A clearly defined set of already implemented, reusable functions.

- **APIs:** A collection of Application Programmer Interfaces (APIs) for accessing COE components. Thus, the COE is a set of building blocks in the same sense that X Windows and Motif are building blocks for creating an application's Graphical User Interface (GUI).

2.3 The DII COE as an Implementation Strategy

The COE is also an evolutionary acquisition and implementation strategy. This represents a departure from traditional development programs. It emphasizes incremental development and fielding to reduce the time required to put new functionality into the hands of the user, while not sacrificing quality nor incurring unreasonable program risk or cost. This approach is sometimes described as a "build a little - test a little - field a lot" philosophy. It is a process of continually evolving a stable baseline to take advantage of new technologies as they mature and to introduce new capabilities. But the changes are done one step at a time so that the warfighters always have a stable baseline product while changes between successive releases are perceived as slight. Evolutionary development has become a practical necessity for many development programs because the traditional development cycle time is longer than the technical obsolescence cycle time.

From the perspective of a COE-based system, the implementation strategy is to field new releases at frequent intervals. Each release might include enhancements to both the COE and mission area applications. Mission-area applications are considered to be provisional, subject to user feedback. Applications for which feedback is favorable are retained in subsequent releases and hardened as needed for continued operational use. As appropriate, mission applications that are widespread in use and commonality will be integrated into the COE or evolved to add new features.

The COE implementation strategy is carefully structured to protect functionality contained in legacy systems so that over time they can migrate to full COE utilization. This is achieved through publishing "public" and "private" APIs. Public APIs are those interfaces to the COE that will be supported for the life cycle of the COE. Private APIs are those interfaces that are supported for a short period of time to allow legacy systems to migrate from unsanctioned to sanctioned APIs. All new development is required to use only public APIs and use of any other APIs results in a non-COE-compliant segment. The process of migrating from existing legacy "stove-pipe" systems to utilizing the COE is a primary source for articulating technical requirements for the COE and it provides program managers with information useful to establishing development priorities.

From the perspective of a system developer, whether developing a new application or migrating an existing one, the COE is an open client/server architecture that offers a collection of services and already built modules for mission applications. Thus, the developer's task is to assemble and customize existing components from the COE while developing only those unique components that are peculiar to particular mission requirements. In many (if not most) cases this amounts to adding new pull down menu entries and icons.

2.4 Assumptions and Objectives

The following assumptions apply to the DII COE:

- The DII COE will migrate to full compliance with the TAFIM standards profile. These standards promote an open systems architecture, the benefits of which are assumed to be well known and generally accepted.

- The DII COE is to be hardware independent and will operate on a range of open systems platforms running under standards-based operating systems. Program driven requirements, associated testing costs, and funding will dictate which specific hardware platforms are given priority.

- Non-developmental items (NDIs), including both commercial off-the-shelf (COTS) and government off-the-shelf (GOTS) products, are the preferred implementation approach.

WWMCCS replacement was the main focus for near-term development, while longer-term development is driven by C4I For the Warrior requirements, logistics support requirements for GCSS, and by financial support requirements for EC/EDI. These broad program drivers lead to a number of program objectives that include those stated in the *TAFIM, Volume 2*:

1. **Commonality**: Develop a common core of software that will form the foundation for Joint systems, initially for C4I and logistics systems.

2. **Reusability**: Develop a common core of software that is highly reusable to take advantage of the investment already made in software development across the services and agencies.

3. **Standardization**: Reduce program development costs through adherence to industry standards. This includes use of commercially available software components whenever possible.

4. **Engineering Base**: Through standardization and an open architecture, establish a large base of trained software/systems engineers.

5. **Training**: Reduce operator training costs and improve operator productivity through enforcement of a uniform human-machine interface, commonality of training documentation, and a consistent "look and feel."

6. **Interoperability**: Increase interoperability through common software and consistent system operation.

7. **Scalability**: Through use of the segment concept and the COE architectural infrastructure, improve system scalability so that COE-based systems will operate with the minimum hardware resources required.

8. **Portability**: Increase portability through use of open systems concepts and standards. This also promotes vendor independence for both hardware and software.

9. **Security**: Improve system security.

10. **Time:** Reduce testing costs because common software can be tested and validated once and then applied to many applications.

SECTION 3

DII ARCHITECTURE AND COE BASELINE CONFIGURATION

This section describes the DII architecture and the DII COE baseline configuration for version 3.1.

3.1 DII ARCHITECTURE

The DII architecture consists of a 3-tier client/server environment incorporating data servers, applications servers, and workstations. These correspond (ideally) to and help support separation of data, function, and presentation services. Each component operates on an 802.3 standard Local Area Network (LAN), dedicated lines, or via dial-up through a communications server. The DII architecture supports a communications capability that provides data transfer facilities among workstations and servers. Figure 3-1 illustrates a notional DII single node infrastructure.

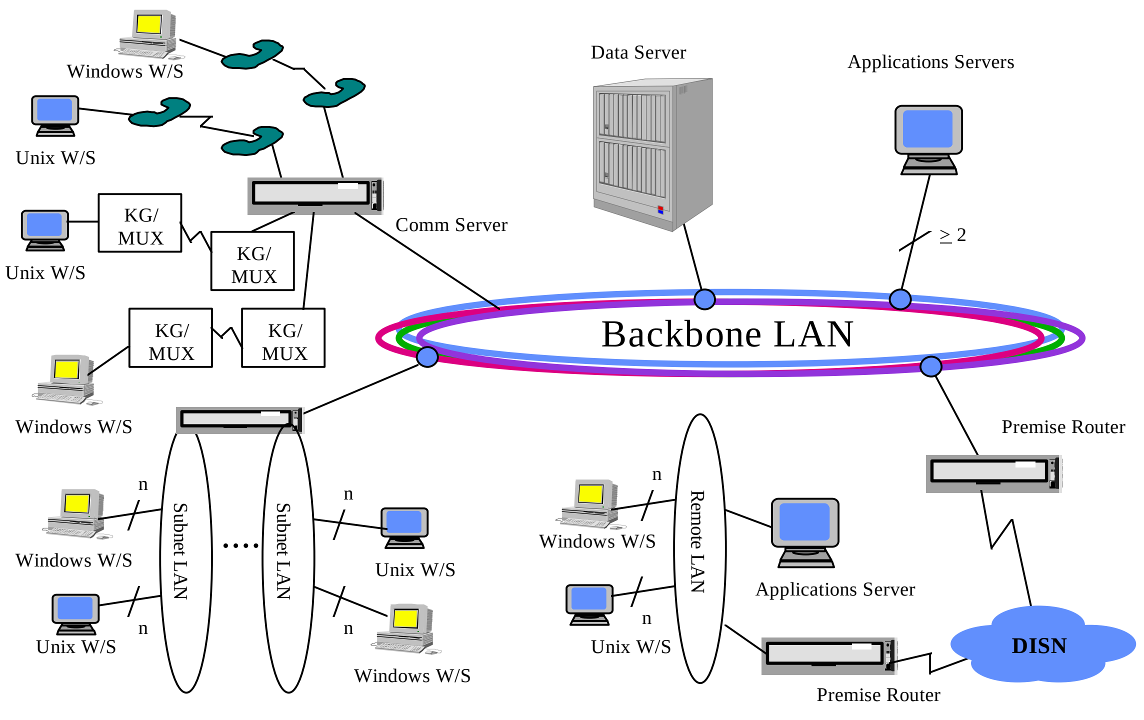

Figure 3-1 DII Notional Single Node Infrastructure

A DII single node infrastructure is, at its simplest, a single workstation but more commonly will consist of at least one database server and two application servers. The database server will act as the repository for all databases and, optionally, applications. The database server may also act as a file server by hosting user accounts, user-specific data, and other site-specific files not deployed with systems. Applications servers host applications and may also act as file servers. One or more subnet LANs may be supported with, optionally, an applications server to localize user account traffic to that subnet. Remote LANs, either logical subnets of the host database site or self-contained, self-administered LANs, can share access to the database server. Finally, remote workstations can access applications and data via dial-up or dedicated circuits. Remotes with limited bandwidth will not have access to the complete suite of mission applications available to local users.

3.2 DII COE VERSION 3.1 BASELINE CONFIGURATION

The following sections represent a consolidated list of all software that is or soon will be available in the DII COE version 3.1 independent of platform. The segments are grouped by COE functionality as it maps to the DII COE taxonomy. For ease of reference, the table in Appendix A provides the current DII COE Taxonomy. In addition, Appendices B and C provide the specific availability of each segment by platform.

3.2.1 Kernel COE Components

3.2.1.1 Operating Systems (OS) and Patches

The operating system load provides a standard release of a vendor-specific OS and a specified set of patches which must be applied to guarantee a standard runtime environment and well-behaved execution of layered COE variants.

3.2.1.2 Desktop

3.2.1.2.1 Common Desktop Environment (CDE). CDE provides a single common desktop for the user, for all applications, across all DII Unix platforms. It provides the desktop interface and other functions provided with it on each workstation.

3.2.1.2.2 Microsoft Windows NT. DII COE on the Microsoft Windows NT platform uses the desktop inherent in Windows NT.

3.2.1.3 Distributed Computing & Object Management

3.2.1.3.1 Distributed Computing Environment (DCE): DCE Client for Solaris. Unlike the Hewlett Packard and Windows NT platforms, Solaris has DCE Client embedded in the kernel.

3.2.1.3.1.1 Threads. The DCE Threads service provides application programmers with the ability to create independent execution threads within the same program. This gives an application the ability to carry out multiple computing tasks concurrently with minimal overhead.

3.2.1.3.1.2 Remote Procedure Call (RPC). The DCE RPC service allows an application component running in one computer to use a simple procedure call mechanism to invoke a procedure running in some other computer on the network. This hides many of the complexities of network communications from application developers.

3.2.1.3.1.3 Distributed Time Service (DTS). The DCE DTS allows application programs to request services that work with date and time-of-day values in a standardized manner that is the same across all computing platforms. The DTS also implements a set of distributed algorithms that ensure the clocks in all the computers in a network are synchronized and contain correct values for the date and time.

3.2.1.3.1.4 Directory Service. The DCE Directory Service implements a distributed repository that stores information about objects in the computing environment, including users, computers, and distributed services that application programs can request. It provides facilities for submitting a name to the Directory Service and getting back a list of the attributes associated with that name. It includes two components; the Global Directory Service and the Cell Directory Service.

3.2.1.4 Printing Services

3.2.1.4.1 Print Services. Print Services provide the basic print capability of the system. It provides such functions as user selection of a default printer, printer administration, and a common way of accessing print resources from an application program. It also includes print queue management and remote printer administration.

3.2.1.5 COE Runtime Tools

The COE Runtime Tools provide basic system administration tools required by the administrator to install, configure, and deinstall systems. They also provide the developers with a means to communicate with the operator during segment installation.

3.2.1.5.1 COEAskUser. Display a message to the user, and have the user click on a button (Yes/No, True/False, Accept/Cancel, etc.) in response to the message.

3.2.1.5.2 COEFindSeg. Return information about a requested segment. The tool sets `status` and writes the pathname, segment name, segment prefix, and segment type information to `stdout`.

3.2.1.5.3 COEInstaller. Display a list of variants or segments that may be installed from tape, disk, or other electronic media. It is normally executed by an operator who selects it from a System Administrator menu to install or deinstall segments.

3.2.1.5.4 COEInstError. Display an error message to the user from within a PreInstall, PostInstall, or DeInstall script signaling termination or de-installation of the segment.

3.2.1.5.5 COEMsg. Display a message to the user and have the user click on the "OK" button to continue. The tool may be used by the PreInstall, PostInstall, and DeInstall scripts.

3.2.1.5.6 COEPrompt. Display a message to the user and have the user enter a response to the message. The tool may be used by the PreInstall, PostInstall, and DeInstall scripts.

3.2.1.5.7 COEPromptPasswd. Prompt user to enter a password. The tool may be used by the PreInstall, PostInstall, and DeInstall scripts.

3.2.1.5.8 COEUpdateHome. Update the home environment variable within a script file to point to where a segment was actually installed.

3.2.1.6 Security Management

In addition to security services, security administration consists of some security-related enhancements to the base runtime environment established by the platform-specific operating system. In particular, the following modules are provided:

3.2.1.6.1.1 Console. Console provides a read-only window for the use of applications which need it (i.e., the application can display information on it but users cannot enter anything into it).

3.2.1.6..1.2 Deadman. Deadman locks the user's terminal if the keyboard and mouse have been idle for longer than a configurable time, defaulting to 5 minutes.

3.2.1.6.1.3 Password. Password allows users to change their own passwords.

3.2.1.6.1.4 X Display Manager (XDM). XDM controls access to the system from the login screen.

3.2.1.6.1.5 Security Manager. Security Manager sets profile configuration, creates or edits local and global user profiles, and creates or edits local and global user accounts.

3.2.1.7 System Mangement

3.2.1.7.1 Process/Session Manager. Manager provides process management.

3.2.1.8 Windowing

3.2.1.8.1 MIT X Windows X11R5. X Windows provides a network-transparent communication protocol between an application and its presentation logic, high-performance device-independent graphics, and a hierarchy of resizable, overlapping windows. This is the standard windowing package for DII Unix-based platforms.

3.2.1.8.2 MOTIF Window Manager. The Open Group's MOTIF is the graphical user interface built on X Windows. This is the standard windowing interface for the DII Unix platforms.

3.2.1.8.3 Microsoft Windows. Microsoft windows is the intrinsic graphical user interface provided with Windows NT. This is the standard windowing interface for the DII Windows NT platforms.

3.2.2 Non-Kernel COE Components

3.2.2.1 Communications

3.2.2.1.1 Unified Build Core (UB). UB Core provides basic Command, Control, Communications, Computers and Intelligence services to receive and process messages, update a track database, perform correlation and data fusion services, and display the tactical picture.

3.2.2.1.2 Link 11/Tadil-A. Link 11/Tadil-A allows the satisfaction of multiple platform requirements using an implementation-independent approach.

3.2.2.1.3 COE Communications Server. COE Communications Server provides the common communications infrastructure for the Army Common Hardware and Software program in support of the Army Battle Command System. The use of the Communications Server software will improve coordination and control of battlefield forces through effective use of communications resource while minimizing program cost through a systematic reuse process.

3.2.2.1.4 Remote Access. Remote Access provides a tool to configure the CISCO 2511 terminal server for PPP access used in support of remote access workstations. It also provides scripts to support the transition from a LAN-based connection to a PPP connection for remote access workstations.

3.2.2.1.5 DII Character Based Interface (CHARIF). CHARIF provides the capability to easily execute character-based applications in a non-graphical menu-driven subsystem using serial and telnet connectivity.

3.2.2.2 Data Access Services

A Relational Database Management System (RDBMS) supplies storage of and access to data objects based on a relational data model. Many of the major RDBMS vendors supplying Unix-based products will be supported across all DII Unix platforms. There are three RDBMSs in the DII COE Version 3.1 baseline and they are as follows:

3.2.2.2.1 Oracle. The Oracle segments provide the database engine, tools, forms, and SQLnet support for DII COE database applications.

3.2.2.2.2 Sybase. The Sybase segments provide the database engine, tools, forms, and SQLnet support for DII COE database applications.

3.2.2.2.3 Informix. The Informix segments provide the database engine, tools, forms, and SOLnet support for DII COE database applications. In addition to databases, Data Management Services include support for the Shared Data Environment.

3.2.2.2.4 Microsoft Access. Access is a database application that allows users to create and manipulate database data.

3.2.2.2.5 DII Database Administration Account Group (DBADM). DII DBADM account group segment provides the database administrator (DBA) with a basic set of services that will manage physical storage, start and stop the Database Management System server, and control the DBA password.

3.2.2.3 Alerts Services

3.2.2.3.1 Alerts. Alerts provides a generic mechanism for the sending and receiving of alert messages between processes.

3.2.2.4 Distributed Computing & Object Management

3.2.2.4.1 Distributed Computing Environment (DCE): DCE Client for Hewlett Packard. Unlike the Solaris platform, Hewlett Packard and Windows NT do not have DCE Client embedded in the kernel.

3.2.2.4.1.1 Threads. The DCE Threads service provides application programmers with the ability to create independent execution threads within the same program. This gives an application the ability to carry out multiple computing tasks concurrently with minimal overhead.

3.2.2.4.1.2 Remote Procedure Call (RPC). The DCE RPC service allows an application component running in one computer to use a simple procedure call mechanism to invoke a procedure running in some other computer on the network. This hides many of the complexities of network communications from application developers.

3.2.2.4.1.3 Distributed Time Service (DTS). The DCE DTS allows application programs to request services that work with date and time-of-day values in a standardized manner that is the same across all computing platforms. The DTS also implements a set of distributed algorithms that ensure the clocks in all the computers in a network are synchronized and contain correct values for the date and time.

3.2.2.4.1.4 Directory Service. The DCE Directory Service implements a distributed repository that stores information about objects in the computing environment, including users, computers, and distributed services that application programs can request. It provides facilities for submitting a name to the Directory Service and getting back a list of the attributes associated with that name. It includes two components; the Global Directory Service and the Cell Directory Service.

3.2.2.4.2 Distributed Computing Environment Servers (DCES)

3.2.2.4.2.1 Security Server. The DCE Security Server provides user Identification/ authentication, user authorization, user access control, and secure data communications services for the DCE aware functions within the cell. Single log-in allows users to log into multiple hosts with a single password. The DCE Security Server is based on Kerberos for identification/authentication and the Portable Open System Interface (POSIX) technologies for access control and audit services. The DCE security service provides a server replication mechanism. If the primary security server is down for any reason, the secondary (or replicated) security server can take over the security operations through the DCE security service administrative commands. The systems hosting a DCE security server must be physically protected from unauthorized access.

3.2.2.4.2.2 Directory Service. The Global Directory Service (GDS) handles directory operations that take place between individual DCE cells. GDS is based on the X.500 standard. The Cell Directory Service (CDS) handles directory operations that take place within a single DCE cell.

3.2.2.4.3 Distributed File Server (DFS). DFS promotes distributed file sharing and management.

3.2.2.4.4 Cell Manager. Cell Manager is a suite of graphical tools that simplifies administration of DCE-based networks. The graphical tools help DCE administrators to organize, monitor, and control access to DCE services.

3.2.2.5 Administrative Services

3.2.2.5.1 File Transfer Protocol Tool (FTPTool). File Transfer Protocol allows users to transfer files between workstations over a network. The FTPTool program provides a GUI for file transfer protocol.

3.2.2.5.2 GZIP. GZIP provides software compression/ decompression services to other segments. Files on disk drives can be stored with less disk space by compressing the files using GZIP. GZIP includes the `gunzip` utility to decompress files compressed by `gzip`, `compress`, or `pack` programs.

3.2.2.5.3 PERL. PERL provides scripting and administration services to other segments.

3.2.2.5.4 Tivoli. Tivoli provides client-server services for managing heterogeneous workstations and other desktop systems.

3.2.2.5.5 News Make Group. allows users to create news groups for use in teleconferencing applications (i.e., News servers for DII).

3.2.2.5.6 NewsPrint. NewsPrint contains the software package (printing drivers) that allow for printing on Solaris.

3.2.2.5.7 Tool Command Language (TCL). Tool Command Language is the interpreter and libraries for the tcl script language and the tk and tcl-x extensions.

3.2.2.5.8 Remote Printing. Remote printing allows users to print at remote workstations.

3.2.2.5.9 Remote Installer. Remote Installer allows users to either push or pull segments from other sources.

3.2.2.6 Security Services

3.2.2.6.1 System Profile Inspector (SPI). SPI is used to allow the examination of the system for its security integrity.

3.2.2.6.2 Crack. Crack has the potential to reveal the passwords on your system. Its use should be restricted to the Security Administrator.

3.2.2.6.3 TCP Wrappers. TCP wrappers is used to monitor and restrict network connections.

3.2.2.6.4 Tripwire. Tripwire is an integrity checker. Tripwire is useful to detect unauthorized changes made by authorized users and to determine what damage your system has sustained after an intrusion.

3.2.2.7 Network Management Services

3.2.2.7.1 HP Openview Network Node Manager (NNM). NNM automatically discovers, maps and monitors enterprise level networks to provide management of distributed multivendor networks and systems.

3.2.2.8 Configuration Management Services

There are no services available as of yet.

3.2.2.9 Multi-Media/Collaborative Services

3.2.2.9.1 Internet Relay Chatter (IRC). IRC is a real-time interactive conferencing tool. Messages input to a conference are made visible to other participants in the conference within seconds.

3.2.2.9.2 Mail Services (MSVCS). MSVCS is a tool that gives the users all components needed to support transmitting electronic mail and any other type of mail functionality.

3.2.2.10 Mapping, Charting, Geodesy & Imagery (MCG&I)

3.2.2.10.1 Joint Mapping Toolkit (JMTK). JMTK provides objects and services to support geospatial analysis, mapping (visual) display, geospatial database management, and image preprocessing.

3.2.2.11 Message Processing

3.2.2.11.1 Common Message Processor (CMP). CMP is the COE's message handling portion of the common support software suite. It provides tools to aid in the preparation and editing of formatted text messages. It also provides normalization software that converts message data into a format usable by the host application software from incoming messages.

3.2.2.12 Office Automation

3.2.2.12.1 Windows Application Based Interface (WABI). WABI emulates the MS Windows environment on the Sun Sparc workstation. WABI allows some MS Windows application programs to be run on the Sun Sparc workstation. Not all MS Windows applications work correctly under WABI. Sun publishes a list of applications known to work.

3.2.2.12.2 Netscape Web Browser. The World Wide Web (WWW) provides users access to other government sites, mailing lists, courseware, educational sites, and commercial sites, to name a few. Netscape provides a graphical tool for accessing the WWW and searching, reviewing, and retrieving information from available sources. It supports electronic mail and access to news groups.

3.2.2.12.3 Netscape News Server. Netscape News Server is the News Server from Netscape. It provides support for hosting news groups on a network.

3.2.2.12.4 NETSITE Server. The Netsite Web Server is a Hypertext Transfer Protocol WWW server.

3.2.2.12.5 Microsoft Office. The Office manager application allows users to manage their MS-Office applications via a floating toolbar.

3.2.2.12.5.1 Microsoft Powerpoint. Powerpoint is a graphics presentation program.

3.2.2.12.5.2 Microsoft Word. Word is a word-processing application.

3.2.2.12.5.3 Microsoft Excel. Excel is a spreadsheet application.

3.2.2.12.5.4 Microsoft Button Bar. Button Bar enables the icon menus on the screen.

3.2.2.12.6 WINDD. WINDD is an application that allows the access to a Windows NT system from a Solaris or HP system.

3.2.2.13 Software Development Services

3.2.2.13.1 Developer's Toolkit

COE Developer's Tools are COE tools which are available during development, but are not delivered to operational sites. All interfaces to these tools are at the command line; none of them has a GUI interface.

3.2.2.13.1.1 CalcSpace. CalcSpace computes the space required for the segment specified and updates the hardware descriptor accordingly. The segment referred to must not be compressed and must not contain any files that do not belong with the segment (e.g., source code) at runtime. The amount of space required is written to `stdout` in K bytes.

3.2.2.13.1.2 CanInstall. CanInstall tests a segment to see if it can be installed, which means that all required segments must already be on the disk and the disk cannot have any conflicting segments.

3.2.2.13.1.3 ConvertSeg. ConvertSeg examines segment descriptors and converts them to the latest format. The original segment descriptor directory is not modified. The output is in a directory created by the tool and called `SegDescrip.NEW`. This directory will be located directly underneath the segment's home directory at the same level as `SegDescrip`.

3.2.2.13.1.4 MakeAttribs. MakeAttribs creates the descriptor file `FileAttribs`. It recursively traverses every subdirectory beneath the segment home directory and creates a file containing permission, owner, group, and filename information.

3.2.2.13.1.5 <u>MakeInstall.</u> MakeInstall writes one or more segments to an installation medium or packages the segments for distribution over the network. MakeInstall checks to see if `VerifySeg` has been run successfully on each of the segments and aborts with an error if it has not.

3.2.2.13.1.6 <u>TestInstall.</u> TestInstall temporarily installs a segment that already resides on disk.

3.2.2.13.1.7 <u>TestRemove.</u> TestRemove removes a segment that was installed by `TestInstall.`

3.2.2.13.1.8 <u>TimeStamp.</u> TimeStamp puts the current time and date into the `VERSION` descriptor.

3.2.2.13.1.9 <u>VerifySeg.</u> VerifySeg validates that a segment conforms to the rules for defining a segment.

3.2.2.13.1.10 <u>VerUpdate</u>. VerUpdate updates the segment version number, date, and time in the `VERSION` descriptor.

3.2.2.13.2 <u>DCE Application Development Tools.</u> These tools are a suite of compilers, library and header files that allow developers to define and manage a set of programs intended to run in a DCE environment. These tools are required to develop software which uses DCE resources.

3.2.2.13.3 <u>Calculator Application (CALC).</u> The CALC Application is intended to support the DCE development environment and consists of one interface with two functions (add and subtract).

	CATEGORY	TWG LEAD	COMPONENTS
Infrastructure Services	Communications Services	Navy	Communications
			Network Services
	Data Access Services	Army	Data Interchange Service
			Database Administration
			Database Management Services
			File Management Services
	Distributed Computing & Object Management Services	Air Force	Distributed Computing Services
	Administrative Services	USAF	System Administration Security Administration
	Network Management Services	DISA	
	Configuration Management Services	DISA	Inventory Control Software Distribution License Management
	Multi-Media/Collaborative Services	Air Force	
	Security Services	DIA	
Support Applications	Common Operational Picture	Navy	
	MCG&I	NIMA	
	Message Processing	Army	Message Processing
			Alerts Service
	Office Automation	Air Force	Office Automation
			On-Line Support
Other	Software Development Services	DISA	HCI Style Guide
			Developer=s Toolkit
			Integration Standards

DII COE KERNEL COMPONENTS

This table provides a detailed list, by platform, of the software segments that are included in the DII COE Version 3.1 Kernel. These segments and services must be loaded on every DII workstation, within a given platform, to be considered Level 5 compliant as defined by the *DII COE Integration and Runtime* Specifications dated January 31, 1997. Table B-1 provides both the DII COE version numbers as well as the COTS version numbers where applicable. Also, please note that not all applications/segments will be released on a single date. For up-to-date release information, please refer to the DII COE Configuration Management Page which can be found on the DII COE Home Page at the following URL:

http://spider.osfL.disa.mil/dii

Functionality	SUN Solaris 2.5.1	Hewlett-Packard UX 10.20	Windows NT 4.0
Kernel	3.0.0.3	3.0.1.0	3.0.0.6
Operating System	103582-01	PHCO_6780 PHNE_6013	

Patches	103594-03 103630-01 103663-01 103680-01 103683-01 103686-01 103743-01 103817-01 DII COE 3.0.0.3P1 DII COE 3.0.0.3P2 DII COE Patch 2 1.0.0.2P2	PHKL_4269 PHSS_5499 PHKL_4334 PHSS_5695 PHKL_6050 PHSS_5696 PHNE_5399 PHSS_6249 DII COE Patch 1 3.0.1.0P1	
Desktop	Common Desktop Environment (CDE) 1.0.0.3/TED 4.0	HP Common Desktop Environment (CDE) 1.0	Inherent
Distributed Computing & Object Management	DCE 1.0.0.1/1.1	N/A	N/A
Printing Services	Print Services 1.0.0.3	Print Services 1.0.0.3	Inherent
Runtime Tools	COEAskUser COEFindSeg COEInstaller COEInstError COEMsg COEPrompt COEPromptPasswd COEUpdateHome	COEAskUser COEFindSeg COEInstaller COEInstError COEMsg COEPrompt COEPromptPasswd COEUpdateHome	COEAskUser COEFindSeg COEInstaller COEInstError COEMsg COEPrompt COEPromptPasswd
Security Management	Console Window 1.2.1.1 Deadman 1.2.1.2 Password 1.2.1.1 XDM 1.2.1.1	Security Services (inherent to HP platform) Console Window 1.2.1.1 Deadman 1.2.1.2 Password 1.2.1.1 XDM 1.2.1.1	Inherent
System Management	Security Manager 1.0	Security Manager 1.0	Inherent
Windowing	Motif 1.0.0.3/1.2.4 X Windows 1.0.0.3/X.11R5	Motif 1.0.0.3/1.2.4 X Windows 1.0.0.3/X.11R5	Inherent

Table B-1 DII COE Version 3.1 Kernel Components

APPENDIX C

DII NON-KERNEL COE COMPONENTS

This table provides a detailed list, by platform, of the DII COE Version 3.1 software segments above and beyond the Kernel. Table C-1 provides both the DII COE version numbers as well as the COTS version numbers where applicable. Also, please note that not all applications/segments will be released on a single date. For up-to-date release information, please refer to the DII COE Configuration Management Page which can be found on the DII COE Home Page at the following URL:

http://spider.osfL.disa.mil/dii

Functional Area	SUN Solaris 2.5.1	Hewlett-Packard UX 10.20	Windows NT 4.0
Communications Services	UB Core 3.0.2.4 Link 11/Tadil-A Interface 2.2.0.2 Link 11/Tadil-A Admin 2.2.0.2 C4I Account Group 1.0.0.1 UB Core 3.0.2.5 Link 11/Tadil-A Interface 2.3.0.0 Link 11/Tadil-A Admin 2.3.0.0 C4I Account Group 1.0.0.3 COE Communications Server 1.4.2.5 Remote Access 1.0.0.1 DII CHARIF 1.0.0.0	UB Core 3.0.2.4 Link 11/Tadil-A Interface 2.2.0.2 Link 11/Tadil-A Admin 2.2.0.2 C4I Account Group 1.0.0.1 UB Core 3.0.2.5 Link 11/Tadil-A Interface 2.3.0.0 Link 11/Tadil-A Admin 2.3.0.0 C4I Account Group 1.0.0.3 DII CHARIF 1.0.0.0	N/A
Data Access Services	Oracle 1.1.0.0/7.3.2.3 Sybase 1.0.0.3/10.0.2a Informix 1.0.0.1/ 7.12 informix 1.0.1.1/7.22 DII DBADM 1.0.0.0	Oracle 1.1.0.0/7.3.2.3 Sybase 1.0.0.2/10.0.2a DII DBADM 1.0.0.0	MS Access 1.0.0.1/2.0
Distributed Computing & Object Management Services	DCES 1.0.0.5/1.1 DCE DFS 1.0.0.0/1.1	DCES 1.0.0.0/1.1 DCEC 1.0.0.0/1.1	N/A
Alerts Services	Alerts 1.3.4.2		
Administrative Services	News Make Group 1.0.0.3 FTP Tool 1.0.0.1 GZIP 1.0.0.1/ 1.2.4 PERL 1.0.0.3/ 5.0.0.2 Tivoli 3.0.0.5 NewsPrint Software 1.0.0.2/2.5 NewsPrint Printer Config 1.0.0.1/2.5 TCL 1.0.0.3/7.4 DII COE Remote Printing 1.0.0.1 DII COE Remote Installer 2.0.0.0 DII COE Component Table 1.0.0.2	News Make Group 1.0.0.3 TCL 1.0.0.3/7.4 GZIP 1.0.0.1/ 1.2.4 PERL 1.0.0.3/ 5.0.0.2 DII COE Remote Installer 2.0.0.0 DII COE Component Table 1.0.0.2	N/A
Security Services	SPI 1.0.0.1/ 3.2.2 Crack 1.0.0.1 TCP Wrappers 1.0.0.2 Tripwire 1.0.0.2/1.2	Crack 1.0.0.1 TCP Wrappers 1.0.0.2 Tripwire 1.0.0.2/1.2	N/A
Network Management Services	HP Openview Network Node Manager 4.11.0.0/4.11 HP NetMetrix Common 4.70.2.0/4.70-002 Domain Manager 4.70.2.0/4.70-002 Internetwork Monitor 4.70.2.0/4.70-02 Mid-Level Manager 4.70.2.0/4.70-002 NetMetrix for NFS 4.70.2.0/4.70-002 Power Agent 4.70.2.0/4.70-002 Reporter 4.70.2.0/4.70-002 Distributed WAN Manager 4.70.2.0/11.01	HP Openview Network Node Manager 4.11.0.0/4.11 HP NetMetrix Common 4.70.2.0/4.70-002 Domain Manager 4.70.2.0/4.70-002 Internetwork Monitor 4.70.2.0/4.70-002 Mid-Level Manager 4.70.2.0/4.70-002 NetMetrix for NFS 4.70.2.0/4.70-002 Power Agent 4.70.2.0/4.70-002 Reporter 4.70.2.0/4.70-002 Distributed WAN Manager 4.70.2.0/11.01 Internetwork Response Agent 4.70.2.0/4.70-002 Internetwork Response Manager 4.70.2.0/4.70-002	N/A
Configuration Management Services	N/A	N/A	N/A
Multi-Media/Collaborative Services	cbif_IRCC 1.0.0.0 IRCC 1.0.0.3/1.16 IRCS 1.0.0.2/2.8.21 MSVCS 1.0.0.2/NA	IRCC 1.0.0.3/1.16 IRCS 1.0.0.2/2.8.21 MSVCS 1.0.0.3/NA	IRCC 1.0.0.0
Mapping, Charting, Geodesy & Imagery	JMTK 1.0.0.9 & 1.0.0.10 NIMA MAP Viewer 1.0.0.0	JMTK 1.0.0.9 & 1.0.0.10 NIMA MAP Viewer 1.0.0.0	N/A
Message Processing	CMP 1.0.2.4 CMP 1.1.0.0	CMP 1.0.2.5	N/A

Table C-1 DII COE Version 3.1 Non-Kernel Components

Functional Area	SUN Solaris 2.5.1	Hewlett-Packard UX 10.20	Windows NT 4.0
Office Automation	Netscape Web Browser 3.0.0.1/3.0 Netscape News Server 1.0.0.3/2.0 NETSITE Server 1.0.0.2/1.1 WABI 1.0.0.3/2.2 WINDD 1.0.0.1/1.0	Netscape Web Browser 3.0.0.0/3.0 Netscape News Server 1.0.0.3/2.0 NETSITE Server 1.0.0.2/1.1 WINDD 1.0.0.1/1.0	Netscape Web Browser 3.0.0.1/3.01 Netscape Web Browser 1.0.0.1/2.0 Adobe Acrobat Reader 1.0.0.0/3.0 Powerpoint 1.0.0.0 Word 1.0.0.0 Excel 1.0.0.1 MS Button Bar 1.0.0.1/4.2
Software Development Services: **DII Developers' Toolkit**	CalcSpace 1.0.0.4 CanInstall 1.0.0.6 ConvertSeg 1.0.0.7 MakeAttribs 1.0.0.7 MakeInstall 1.0.1.5 TestInstall 1.0.0.7 TestRemove 1.0.0.6 TimeStamp 1.0.0.6 VerfySeg 1.0.0.7 VerUpdate 1.0.1.5 Secomp 1.0.0.3	CalcSpace 1.0.0.4 CanInstall 1.0.0.6 ConvertSeg 1.0.0.7 MakeAttribs 1.0.0.7 MakeInstall 1.0.1.5 TestInstall 1.0.0.7 TestRemove 1.0.0.6 TimeStamp 1.0.0.6 VerfySeg 1.0.0.7 VerUpdate 1.0.1.5 Secomp 1.0.0.3	CalSpace 1.0.0.4 CanInstall 1.0.0.6 MakeInstall 1.0.1.5 TestInstall 1.0.0.7 TestRemove 1.0.0.6 TimeStamp 1.0.0.6 VerfySeg 1.0.0.7 VerUpdate 1.0.1.5
DCE Developer's Toolkit	DCE Apps Dev Tools 1.0.0.0 CALC 1.0.0.0	DCE Apps Dev Tools 1.0.0.0 CALC 1.0.0.0	
Unified Build Developer's Tools	UB Developer's Tools 3.0.2.4 & 3.0.2.5	UB Developer's Tools 3.0.2.4 & 3.0.2.5	
Joint Mapping Toolkit Developer's Tools	JMTK Tools 1.0.0.9 & 1.0.0.10	JMTK Tools 1.0.0.9 & 1.0.0.10	

Table C-1 Continued

Defense Information Infrastructure (DII) Common Operating Environment (COE)

Version 3.1

Baseline Specifications

April 29, 1997

Approved by:

____

DAWN A. HARTLEY
DII COE Chief Engineer
Software/Data Architecture Engineering Division
Center for Computer Systems Engineering